VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

CredentialProvider pro přihlašování uživatele do Windows PC

CredentialProvider that allows users to login into Windows PC

2015 Lukáš Fila

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Fila**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: CredentialProvider pro přihlašování uživatele do Windows PC
CredentialProvider that Allows Users to Login into Windows PC

Zásady pro vypracování:

Vývoj modulu Credential Provider, který umožní přihlášení uživatele do Windows Vista - 8.1 pomocí: jména a hesla uloženého na čipové kartě (i bezkontakrní) a certifikátu X.509 a privátního klíče, které jsou uloženy v obecném CSP (Cryptographic Service Provider).

1. Studium technologie CredentialProviders, která je podporována ve Windows Vista a vyšších.
2. Analýza rozdílů rozhraní mezi verzemi Windows.
3. Modely využití externích čipových karet, návrh procesů správy přihlašovacích údajů na čipových kartách.
4. Analýza problematiky životního cyklu přihlašovacích údajů (změna hesla, reset hesla, ...).
5. Návrh CredentialProvider pro podporu použití čipových karet a jména/hesla (WIndows Vista - 8.1).
6. Vývoj CredentialProvider pro podporu použití čipových karet a jména/hesla (WIndows Vista - 8.1).
7. Návrh CredentialProvider pro podporu použití CSP a X.509 s privátním klíčem (WIndows Vista - 8.1).
8. Vývoj CredentialProvider pro podporu použití CSP a X.509 s privátním klíčem (WIndows Vista - 8.1).
9. Návrhy dalšího rozvoje technologie.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Milan Hrdlička**

Konzultant bakalářské práce: Ing. Pavel Moravec, Ph.D.

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

Veškeré zdrojové kódy vzniklé pro tuto práci jsou majetkem firmy Monet+ a.s. a zůstanou neveřejné.

V Ostravě 7. května 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, zejména inženýru Milanu Hrdličkovi za vedení, doktoru Pavlovi Moravcovi za pomoc a rady a také doktoru Jiřímu Gieslovi za povzbudivá slova a radu v okamžicích, kdy jsem se ocitl ve slepých uličkách.

Abstrakt

Systémy Windows Vista a vyšší podporují modulární architekturu implementace různých přihlašovacích metod. Ta je založena na tzv. Credential Providers - na modulech, jejichž zodpovědností je komunikace s uživatelem během přihlašovacího procesu a zajištění přihlašovacích údajů. Oproti uzavřené předchozí technologii GINA jsou Credential Providers oficiálně podporovaným rozhraním pro rozšiřování funkcí operačního systému. Standardní instalace Windows podporuje jen omezené portfolio přihlašovacích metod (ručně zadané jméno/heslo, čipová karta s certifikátem X.509, PIN a základní podpora pro biometrii). Cílem zadání této bakalářské práce je vytvoření přihlašovacích modulů, které umožní použít čipovou kartu spolu se jménem a heslem a obecné úložiště certifikátů X.509, které není vázáno na čipové karty.

Klíčová slova: Credential Provider, MS Windows, Přihlášení, CSP, X.509

Abstract

MS Windows systems Vista and later provide modular architecture for implementation of different authentication methods. The architecture is based on Credential Providers - modules for interaction with user in authentication process and obtaining his credentials. Compared to previous GINA method the Credential Providers are officially supported interface for expanded functionality of OS. Standard installation of Windows provides only a limited list of authentication methods (manualy entered login/password, Smart card with X.509 certificate and PIN and basic support for biometric technology). The aim of this thesis is creation of authentication modules, which allow the use of smart cards with password and login and general storage of X.509 certificate, that is not tied to a smart card.

Keywords: Credential Provider, MS Windows, Authentication, CSP, X.509

Seznam použitých zkratek a symbolů

AD	–	Active Directory
AD DS	–	Active Directory Domain Services
AES	–	Advanced Encryption Standard
APDU	–	Application Protocol Data Unit
API	–	Application Programming Interface
ASN.1	–	Abstract Syntax Notation One - zápis binárních dat
CNG	–	Cryptography API: Next Generation
COM	–	Component Object Model
CP	–	Credential Provider
CRL	–	Certificate Revocation List
CSP	–	Cryptography Service Provider
DES	–	Data Encryption Standard
DLL	–	Dynamic Load Library
EMV	–	Standard pro platební karty
EV	–	Evoluce - označení novějších karet MIFARE
EXE	–	Executable - spustitelný soubor
GINA	–	Graphical Identification and Authentication
KSP	–	Key Service Provider
LSA	–	Local Security Authority
MIFARE	–	typ bezkontaktního čipu od firmy NXP Semiconductors
MS	–	Microsoft
NFC	–	Near Field Communication - bezkontaktní technologie
OS	–	Operační systém
PC	–	Personal Computer
PC/SC	–	Standard pro komunikaci mezi PC a SC
PGP	–	Pretty Good Privacy - Šifrovací standard a program
PIN	–	Personal Identification Number
PKI	–	public Key Infrastructure
PKINIT	–	Public Key Cryptography for Initial Authentication in Kerberos
PUK	–	Personal Unblocking Code
RFID	–	Radio Frequency Identification
SAS	–	Security Atention Sequence

SC	–	Smart Card
SDK	–	Software Development Kit
SHA-1	–	Secure Hash Algorithm
SID	–	Security Identifier
SIM	–	Subscriber Identity Module
TLV	–	Tag Length Value - formátování zápisu dat
USB	–	Universal Serial Bus - sérioví komunikační rozhraní
X.509	–	Kryptografický standard pro PKI systémy

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce popisuje využití čipových karet a certifikátů pro přihlášení do systému Windows. Cílem práce je pochopení způsobu přihlašování do systému Windows a způsob jeho implementace pomocí technologie Credential Provider (CP). K implementaci CP je použito rozhraní distribuované v Credential Provider SDK, určené pro jazyk C++.

CP umožňuje firmám, které se zabývají bezpečností a autentizací, aby mohly nabídnout svým zákazníkům řešení pro autentizaci do systému šité na míru. Mohou rozšiřovat standardní řešení od firmy Microsoft o další faktory, a tím zvýšit bezpečnost přihlášení.

Důvodem vzniku této práce je prozkoumání technologie Credential Provider a vytvoření konceptu autorizace uživatelů pomocí certifikátů a klíčů uložených mimo fyzickou čipovou kartu. Firma Monet+ a.s. nyní nabízí svým zákazníkům autorizační server CASE, to bude v budoucnu využit pro vytvoření virtuálních karet, které budou tento server využívat jako své úložiště dat.

Hlavním cílem práce je ověřit, zda lze provést autentizaci za použití certifikátu s privátním klíčem uloženým mimo čipovou kartu a použití standardních komponent systému MS Windows. Pro přístup k certifikátům je používán tzv. Cryptography Service Provider (CSP) a pro přístup k jeho privátnímu klíči je použita technologie Key Service Provider (KSP). Pokud nelze tohoto cíle dosáhnout za použití standardních prostředků firmy Microsoft, je potřeba navrhnout nutné rozšíření.

Obsahem této práce je rozbor technologie Credential Provider. Další úkol je ověřit pochopení této technologie v podobě implementace Credential Provider, který umožňuje přihlášení pomocí uživatelských údajů uložených na čipové kartě. Je nutné vyřešit problémy při správě účtů na čipové kartě, zajištění shodnosti uložených údajů na kartě s údaji v systému, problémy s přidávání a odebírání účtů v systému a na kartě, řešení zapomenutí hesla uživatelem.

Následuje analýza procesu přihlášení za použití certifikátu na čipové kartě. Z toho poté vyplývají nutné úpravy pro implementaci přihlášení s certifikátem a klíčem uložených mimo čipovou kartu.

2 Přehled použitých technologií

2.1 Čipová karta

Čipová karta je plastová karta o velikosti zpravidla cca 86 x 54 mm, definované v ISO 7810, která je osazena čipem s kontaktním rozhraním dle ISO 7816 či bezkontaktním rozhraním ISO 14443. Existují karty mající obě rozhraní. Tyto karty se rozlišují na hybridní a duální. Hybridní karta má pro každé rozhraní vlastní čip. Duální karta je naopak taková, která má jeden čip, se kterým jde komunikovat po obou rozhraních. Zde nastává problém pokud, se snažíme komunikovat na obou rozhraních zároveň. To se stává u duálních čteček, do kterých vložíme kartu do kontaktní části a systém na připojeném PC začne automaticky komunikovat i na bezkontaktním rozhraní (detekce karty a instalace ovladače).

2.1.1 Historie

První karta byla patentovaná v letech 1968 a 1969 dvojicí německých elektroinženýrů Helmutem Gröttrupem a Jürgenem Dethloffem. V roce 1978 pak Jürgen Dethloff patentoval první čipovou kartu s mikroprocesorem a pamětí, kterou vyvinul v laboratořích Honeywell. První průmyslově vyráběná karta, založena na tomto patentu, byla CP8 od firmy Motorola. Největší nárůst použití čipových karet přišel s nástupem používání SIM karet s GSM technologii v roce 1990. Používání karet širokou veřejností přišlo s nástupem platebním systémem EMV roku 1994.

2.1.2 Využití

Hlavní využití čipových karet je v různých službách, které vyžadují autentizaci. Hlavní kategorie těchto služeb jsou finanční služby (platební karty - platba na terminálech nebo přes Internet, elektronická peněženka), komunikační služby (satelitní karty Irdeto či CryptoWorks, SIM karty do GSM sítí), informační bezpečnost (přístup do sítí), řízení přístupů (zaměstnanecké karty, může být spojeno s biometrií), doprava (tachograf, karty na MHD či vlak), věrnostní karty.

2.2 Rozhraní PC/SC

Rozhraní PC/SC je rozhraní pro komunikaci mezi počítačem (PC) a čipovou kartou (SC), které unifikuje přístup ke čtečkám a čipovým kartám. Programátor je odstíněn od funkcí čtečky, jako nastavení napětí, určení rychlosti a frekvence přenosu, a používá jen omezenou sadu funkcí, které pracují stejně na různých typech čteček. Implementace firmy Microsoft tohoto rozhraní je WinSCard API. V unixovém světe je nejrozšířenější implementace PC/SC Lite. Sjednocuje i APDU protokol pro kontaktní i bezkontaktní karty, kdy pro bezkontaktní se používá zapouzdření bezkontaktních APDU (dle ISO 14443) na formát kontaktních APDU (dle ISO 7816). Zde pak záleží na konkrétní čtečce, zda si daný příkaz odbalí a pošlé ho kartě v standardní podobě. To vše je však programátorovi skryto.

2.3 Bezkontaktní karty

2.3.1 Technologie RFID

Bezkontaktní karty jsou založeny na bázi RFID technologie. Je to moderní technologie na identifikaci objektů pomocí rádiových vln. Jde o velmi rychlou a přesnou metodu zpracování informací. Na začátku vzniku této technologie stála firma WalMart, která chtěla nahradit nejpoužívanější technologii pro identifikaci objektů – čárový kód. Základní nevýhoda čárového kódu je jeho optické čtení, což znamená, že kód musí být přímo viditelný pro jeho zpracování. RFID vyžaduje jen relativní blízkost objektu a moderní čtečky zvládají přečíst dokonce několik stovek kusů za minutu. Tato technologie našla uplatnění především v logistice a to pro sledování logistických jednotek (zboží, paleta, kontejner), sledování kufrů na letišti či sledování osob (přístupové RFID klíče).

Z hlediska komunikace dělíme RFID tagy na aktivní a pasivní. Aktivní mají uvnitř vlastní zdroj napětí v podobě baterie a samy tak vysílají data bez vyžádání. Mají zato ale vyšší cenu a nižší životnost. Pasivní tagy se napájejí pomocí elektromagnetické indukce – čtečka vyšle elektromagnetické vlny, které se zachytí na anténě RFID tagu. Tím se vytvoří elektrické napětí zachycené na malém kondenzátoru uvnitř tagu a z tohoto kondenzátoru se napájí vlastní čip.

2.3.2 Historie karet

Na bázi technologie RFID vznikla norma ISO 14443, která určuje mezinárodní standardy pro bezkontaktní čipové karty využívané pro autentizaci a protokol ke komunikaci s těmito kartami. Pro komunikaci je využit nosný signál o frekvenci 13,56 MHz. Dle způsobu modulace signálu na nosném signálu rozlišujeme čipy typu A a B. V současnosti většina čipů komunikuje protokolem typu A (MIFARE, JCOP, MPCOS).

Každý vyrobený čip má své unikátní sériové číslo UID, dlouhé 4 nebo 7 bajtů. Toto číslo většinou leží v prvním bloku paměťového prostoru karty a je read-only.

Kolem roku 2005 začaly být vydávány bezkontaktní karty pro bez-pinové platby malých částek. Jsou dva základní typy bezkontaktních platebních karet. První typ je jen malá paměť, která obsahuje ekvivalentní data k magnetickému proužku, a druhý je plnohodnotná EMV karta.

2.3.3 MIFARE

Jednou z nejpoužívanějších skupin bezkontaktních čipů jsou čipy MIFARE společnosti NXP Semiconductors (dceřiná společnost firmy Philips). Dělí se na dva základní typy karet.

První jsou paměťové karty, které jsou levnější, a nabízí jen klíčem zabezpečený úložný prostor. První čip byl MIFARE classic 1k a 4k, obsahující 1k či 4k bajty paměťového prostoru. Jsou NFC kompatibilní. Kvůli kompromitaci jejich kryptografie byly nahrazeny čipy MIFARE Ultralight (EV1, C), čipy s lepším zabezpečením, ale menší velikostí paměťového prostoru 48, 64, 128 či 196 bajty .

Čipy DESFire, které obsahují obecně použitelný MIFARE DESFire operační systém, nabízející základní adresářovou a souborovou strukturu. Jsou založeny na procesoru 8051 s 3DES/AES koprocesorem, což jim umožňuje velmi rychlé provádění transakcí. Po kompromitaci jejich bezpečnosti byla ukončena jejich podpora a nastoupily novější varianty EV1 a EV2.

2.4 Public Key Infrastructure

Public Key Infrastructure (PKI) je sada hardwarových a softwarových politik a procedur pro vytváření, správu, distribuci, používání, uložení a zneplatnění digitálních certifikátů a správa veřejných šifrovacích klíčů. Hlavní cíl PKI je zabezpečit elektronický přenos důvěrných dat přes nedůvěryhodné médium (např. Internet) a jednoznačné ověření identity protistrany. Data jsou chráněna pomocí asymetrické kryptografie. Privátní klíč si drží subjekt na bezpečném úložišti (čipová karta, HSM modul) a veřejný klič je distribuován v obálce, dle které se jednoznačně identifikuje držitel klíče, typ klíče, jeho povolené použití a lze ověřit jeho platnost a důvěryhodnost. Takováto obálka se nazývá certifikát.

Součástí každého certifikátu je digitální podpis pro ověření jeho pravosti. Tento podpis vzniká zašifrováním dat privátním klíčem. A to buď klíčem na, který odkazuje certifikát (self-signed), anebo klíčem nadřazené autority. Tomuto nadřazenému subjektu se pak říká vystavitel.

Existuje dvojí způsob důvěryhodnosti certifikátů. Pomocí důvěryhodných certifikačních autorit, anebo pomocí sítí důvěry. V případě certifikačních autorit se jedná o hierarchickou strukturu, kdy existuje autorita, které všichni věří, a která dále vydávají certifikát mezilehlým autoritám, či přímo koncovému klientu. Při ověřování klientského certifikátu musíme najít řetězec certifikátů mezi ním a důvěryhodným certifikátem autority. A to tak, že podle veřejného klíče z certifikátu vystavitele ověříme digitální podpis v samotném certifikátu. Takto dojdeme až k certifikátu autority, který jediný je podepsaný sám sebou – identifikátor subjektu je stejný jako identifikátor vystavitele. Nejrozšířenější implementace PKI tohoto typu je X.509 standard.

Síť důvěry je decentralizovaný přístup k důvěryhodnosti subjektů. Každý subjekt je si sám sobě certifikační autoritou. Většinou má dva páry klíčů, jeden na šifrování a druhý na podepsání tohoto šifrovacího klíče v certifikátu. Každý subjekt má vlastní seznam certifikátů a ten seznam veřejně distribuuje. Každému certifikátu na svém seznamu přidělí subjekt určitý stupeň důvěry. Když pak příjde na řadu ověření důvěryhodnosti nového certifikátu, subjekt projde seznamy důvěryhodných subjektů. Pokud některý z nich tomuto certifikátu již plně věří, či pokud více subjektů věří certifikátu částečně, je pro subjekt certifikát taky důvěryhodný. Autor tohoto konceptu je Phil Zimmermann, který ho sám také implementoval v podobě programu PGP.

Rozhodování o důvěryhodnosti jednotlivých certifikátů má sám uživatel. S tím souvisí i jeho odpovědnost. Při neopatrném rozhodování může narušit bezpečnost celé sítě důvěryhodnosti.

Naproti tomu u X.509 je uživatel nucen plně věřit certifikátům autorit, na jejichž plné odpovědnosti je správa těchto certifikátů. Uživatel si kromě smazání tohoto kořenového

certifikátu nemůže jinak snížit stupeň důvěryhodnosti. Další důležitý rozdíl je v tom, že jeden certifikát je podepsán pouze jednou certifikační autoritou.

2.4.1 X.509

Nejrozšířenější PKI standard hierarchického type je X.509 technologie. Definuje mimo jiné standardy pro certifikáty veřejných klíčů, revokační seznam certifikátů, parametry certifikátů a algoritmy pro kontrolu platnosti certifikátu. Jednotlivé certifikační autority na žádost držitele privátního klíče vygenerují certifikát, který váže jeho rozlišující název na konkrétní veřejný klíč. Každý certifikát obsahuje několik povinných a nepovinných položek. Mezi povinné patří sériové číslo certifikátu, verze X.509, identifikátor algoritmu klíče, název vystavitele, doba platnosti, název subjektu a samotný veřejný klíč. Mezi volitelné parametry patří unikátní identifikátor vystavitele a subjektu (bývá to hash veřejného klíče) a rozšíření, jako například omezení použití (autentizace, šifrování, podepisování, aj.).

Důležitou roli při ověřování řetězce certifikátu hraje CRL. Při každém ověření certifikátu by se mělo dohledat, že daný certifikát není revokovaný jeho vystavitelem. Tyto revokační seznamy mají většinou dobu životnosti v řádu desítek hodin a je nutné je často stahovat. Distribuční bod CRL je obsažen jako parametr ve vlastním certifikátu autority. Všechny tyto operace výrazně zvyšují náročnost na ověření platnosti jednotlivých certifikátů.

2.5 Přihlášení do systému Windows

Od systému Vista začala firma Microsoft používat odlišnou metodu přihlašování oproti předchozí technologii GINA. Ta byla pro vývojáře hůře pochopitelná a náročná ze strany firmy Microsoft na údržbu. Nová technologie zvaná Credential Provider je navržená jako modulární, plug-in rozhraní, pomocí kterého si programátor může napsat vlastní GUI pro komunikaci s uživatelem, a integrovat ty autentizační technologie, které nejlépe vyhovují cílovému uživateli (biometrické senzory, NFC tagy, apod.).

Další rozdíl od technologie GINA je ve způsobu volání přihlašovacího modulu od systému. GINA modul byl spouštěn v jednom sezení s ostatními službami, jako LSASS, což snižuje celkovou bezpečnost systému. Za to od systému Vista je kladen důraz na oddělení služeb, které běží jednou pro celý stroj (LSASS apod.), a služeb, které běží pro každého uživatele ve vlastní instanci jako je CP.

Samotný přihlašovací proces je složen z následujících komponent (viz Tabulka 1).

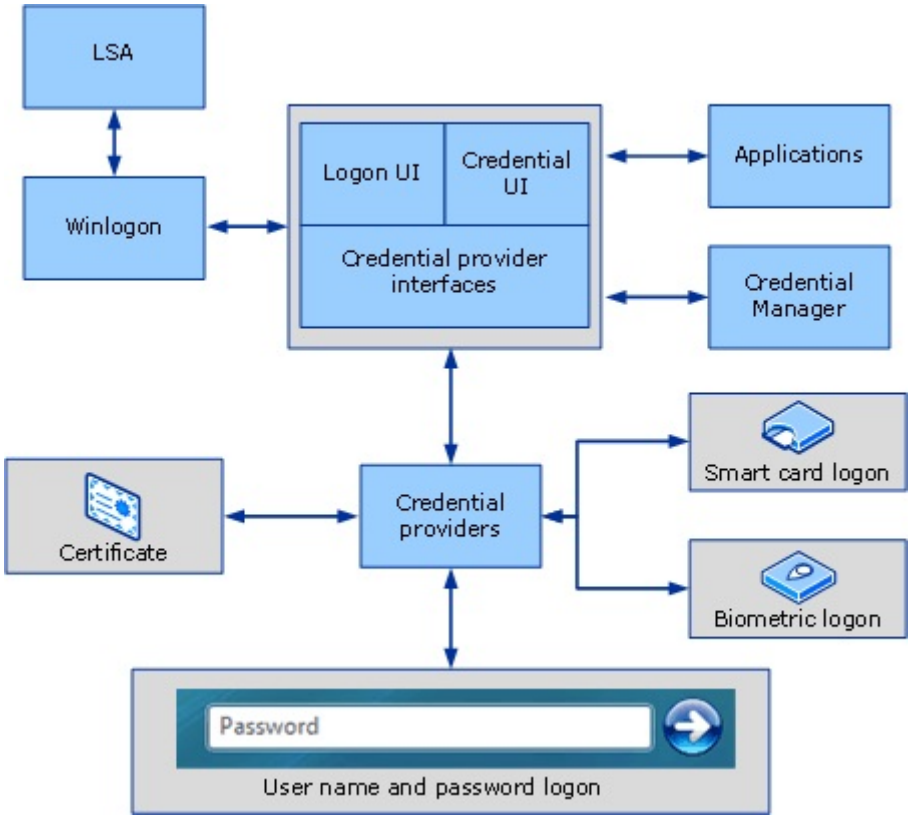
Přihlašovací proces začíná zpravidla stisknutím klávesové kombinace CTRL+ALT+DEL. Této kombinaci se říká bezpečnostní upozorňovací sekvence (secure attention sequence). Aby se zabránilo používání této kombinace jinými programy, winlogon.exe si ji zaregistruje již během startu systému. Logon UI vykreslí názvy a obrázky jednotlivých credentialů, které získá od zaregistrovaných CP. Obvykle musí uživatel, který se chce přihlásit na lokální či doménový účet, zadat uživatelské jméno a heslo. Tyto údaje jsou použity k ověření identity uživatele. Při použití čipových karet jsou bezpečně uloženy v paměti čipu a při přihlášení zadá uživatel jen přístupový kód ke kartě (PIN) místo jména a hesla.

Samotné CP jsou aplikační COM objekty, které slouží k sesbírání uživatelových přihlašovacích údajů, a běží v lokálním sezení počítače. Logon UI komponenta poskytne prostředky pro vykreslení GUI, winlogon poskytne prostředky pro interaktivní přihlášení a CP pracuje s oběma komponentami pro sběr a zpracování přihlašovacích údajů.

Po stisknutí upozorňující sekvence winlogon pošle logon UI požadavek na vykreslení dlaždic. Získá je tak, že se zeptá každého registrovaného CP na počet jednotlivých credentials, které chce zobrazit. Každý provider může poté říct, zda považuje jeden cre-

Komponenta	**Popis**
Winlogon	Poskytuje infrastrukturu pro interaktivní přihlášení
Logon UI	Poskytuje vykreslování GUI komponent
Credential Providers	Vypisují přihlašovací informace a serializují přihlašovací údaje
LSA	Zpracovává přihlašovací údaje
Authentication packages	Nese uživatelovy autentizační údaje k serveru

Tabulka 1: Tabulka autentizačních komponent (zdroj: http://www.msdn.microsoft.com)



Obrázek 1: Struktura přihlašovacích komponent
(zdroj: http://www.msdn.microsoft.com)

dential ze svého seznamu za základní, který by se měl přednostně použít. Pokud více providerů zadá svůj základní credential a žádný z nich nemá příznak auto-login, použije se základní credential z provideru, pomocí kterého byl poslední uživatel přihlášen, a zobrazí se jenom tento základní credential. Pokud žádný credential nemá prioritu jako základní, logon UI zobrazí seznam všech credential. A poté co uživatel zadá své přihlašovací údaje, je logon UI poskytne pro autentizaci.

Společně s patřičným hardwarem můžou CP rozšířit systém Windows o autentizaci pomocí biometrických údajů (např. otisk prstu, rozpoznání hlasu). Velké společnosti a bezpečnostní experti mohou vyvinout svůj vlastní přihlašovací mechanismus pro všechny své doménové uživatele. Mohou dokonce striktně vynutit využívání jejich CP.

CP samy o sobě neprověřují přihlašovací údaje, jen je sbírají. LSA s pomocí přihlašovacích balíčků provádí samotnou autentizaci.

CP API nevykresluje GUI. Jen poskytuje popis toho, co se má vykreslit. V safe módu je k dispozici pouze přihlášení pomocí hesla. Přihlášení pomocí čipové karty je k dispozici v safe módu s podorou sítě.

2.5.1 Credential Provider

API pro vývoj CP je dodáváno v SDK pro jazyk C++. Pro překlad je potřeba použít nástroj Visual Studio 2005 a novější. Pro vývoj CP v1 je nutné Windows SDK v minimální verzi 6. Pro CP v2 je nutné Windows SDK od verze 8.

Hlavní funkcí CP je získání přihlašovacích údajů uživatele, jejich zpracování a odeslání Kerberosu k ověření.

Každý CP se skládá ze dvou základních částí – provideru a credentialu. Každý CP má právě jednu instanci třídy odvozené od rozhraní *ICredentialProvider*, jejíž jednotlivé metody volá systém pro získání informací. Mezi tyto informace patří případy užití, které CP implementuje, počet jednotlivých credentials, pointer na jednotlivé credentials, počet grafických prvků v credentials, typ a popis jednotlivých grafických prvků a registrovaný callback pomocí kterého informují externí prvky provider o změně stavu (např. vložení čipové karty, apod.).

Druhou částí jsou jednotlivé credentials. Jsou to instance tříd odvozených od rozhraní *ICredentialProviderCredential*, kdy jeden provider může obsahovat více různých tříd credentials. Přičemž aktivní bývají jen instance jedné třídy, a to hlavně kvůli tomu, že počet, pořadí a typy grafických prvků jsou definovány jako společné pro celý provider (může se ale dynamicky měnit na pokyn nějakého externího prvku). Každá aktivní instance credentialu představuje pro uživatele jeden box (dlaždici) zobrazený na obrazovce.

Každý grafický prvek v credentialu má nastaven atributy pro viditelnost (viz Tabulka 2) a interaktivitu. Momentálně je podporován jen příznak interaktivity *FOCUSED*, který určuje ve kterém prvku je editační kurzor. Zvláštní grafický prvek je tlačítko *SUBMIT*, viditelné pouze ve vybraném stavu credential, pomocí kterého uživatel dává systému znamení, že již vyplnil veškeré údaje.

Každá třída implementující rozhraní *ICredentialProviderCredential* obsahuje metody, pomocí kterých si sytém bere informace o jednotlivých prvcích, a pracuje se zadanými přihlašovacími údaji. Je to sada metod na získání údaje prvku (text, obrázek, binární

Název	Popis
HIDDEN	Prvek se nezobrazuje
IN_SELECTED_TILE	Prvek se zobrazí jen pokud je credential vybrán
IN_DESELECTED_TILE	Prvek se zobrazí jen v základním zobrazení
IN_BOTH	Prvek se zobrazí v obou případech

Tabulka 2: Tabulka stavu grafických prvků

Použití	Popis
LOGON	Přihlášení do systému
UNLOCK_WORKSTATION	Odemknutí počítače
CHANGE_PASSWORD	Změna hesla či pinu
CREDUI	Grafická výzva na zadání přihlašovacích údajů například přístup na SMB disk
PLAP	Přístup před přihlášením

Tabulka 3: Tabulka použití CP

informace checkboxu, počet prvků v comboboxu, jednotlivé prvky v comboboxu, umístění submit tlačítka), změna údajů od uživatele (text, hodnota checkboxu, vybraný prvek comboboxu), zabalení přihlašovacích údajů pro Kerberos, interpretace výsledku, registrace callbacku pro aktualizaci stavu credentialu a reakci na vybrání/odvybrání daného credential. Každý credential je graficky reprezentován dlaždicí obsahující obrázek a popisek (většinou uživatelské jméno) v nevybraném stavu. Ve vybraném stavu se zobrazí i ty prvky, které slouží k zadání hodnot.

Každý CP má definovanou množinu použití (viz Tabulka 3).

Jeden CP může implementovat chovaní pro více použití. Výjimkou je *PLAP*, který musí být samotný. Pokud je implementován s například se scénářem *LOGON*, vyvolá se jen samotné přihlášení uživatele a ne předpřihlášení.

Systém Windows 8 přinesl s sebou rozšíření v podobě CP v2, kdy je každému providerú poskytnut i seznam uživatelů, pro které se žádají credentials. Systém poté sloučí všechny možné credentials od různých providerů pro jednoho uživatele do jednoho okýnka se jménem uživatele. Uživatel si nejprve vybere systémového uživatele, pod kterým se chce přihlásit, a až poté si volí způsob autentizace. Dále podporuje větší obrázek CP 200x200 px místo 126x126 px. Při použití CP v1 na systému Windows 8 je obrázek orámován šedým pozadím.

3 Popis funkcí rozhraní Credential Provideru

Credential Provider je DLL knihovna napsaná v C++, která implementuje dvě třídy reprezentující provider a výčet jeho credentials.

3.1 Provider

Provider musí implementovat rozhraní *ICredentialProvider 1* a v případě v2 i rozhraní *ICredentialProviderSetUserArray 2*. Tyto rozhraní obsahují metody, pomocí kterých winlogon komunikuje s CP a získává data o jednotlivých dlaždicích.

```
ICredentialProvider : public IUnknown
{
 public:
    virtual HRESULT STDMETHODCALLTYPE SetUsageScenario(
        /* [in] */ CREDENTIAL_PROVIDER_USAGE_SCENARIO cpus,
        /* [in] */ DWORD dwFlags) = 0;

    virtual HRESULT STDMETHODCALLTYPE SetSerialization(
        /* [in] */ const CREDENTIAL_PROVIDER_CREDENTIAL_SERIALIZATION *pcpcs)
            = 0;

    virtual HRESULT STDMETHODCALLTYPE Advise(
        /* [in] */ ICredentialProviderEvents *pcpe,
        /* [in] */ UINT_PTR upAdviseContext) = 0;

    virtual HRESULT STDMETHODCALLTYPE UnAdvise( void) = 0;

    virtual HRESULT STDMETHODCALLTYPE GetFieldDescriptorCount(
        /* [out] */ DWORD *pdwCount) = 0;

    virtual HRESULT STDMETHODCALLTYPE GetFieldDescriptorAt(
        /* [in] */ DWORD dwIndex,
        /* [out] */ CREDENTIAL_PROVIDER_FIELD_DESCRIPTOR **ppcpfd) = 0;

    virtual HRESULT STDMETHODCALLTYPE GetCredentialCount(
        /* [out] */ DWORD *pdwCount,
        /* [out] */ DWORD *pdwDefault,
        /* [out] */ BOOL *pbAutoLogonWithDefault) = 0;

    virtual HRESULT STDMETHODCALLTYPE GetCredentialAt(
        /* [in] */ DWORD dwIndex,
        /* [out] */ ICredentialProviderCredential **ppcpc) = 0;

};
```

Výpis 1: Rozhraní ICredentialProvider

```
ICredentialProviderSetUserArray : public IUnknown
{
public:
    virtual HRESULT STDMETHODCALLTYPE SetUserArray(
```

```
        /* [in] */ ICredentialProviderUserArray *users) = 0;

};
```

Výpis 2: Rozhraní ICredentialProviderSetUserArray

SetUsageScenario
Tato metoda slouží k definici jednotlivých scénářů použití. Systém zjišťuje, zda CP podporuje daný scénář. Typ scénáře vstupuje jako první proměnná, jeho jednotlivé příznaky jsou druhá vstupní proměnná. Jednotlivé scénáře jsou popsané v Tabulce 3. V případě, že CP daný scénář podporuje, vrací S_OK. V případě, že nepodporuje, měl by vrátit *E_NOTIMPL*. V případě chyby při inicializaci scénáře by měl CP tuto chybu vrátit.

SetSerialization
Tato metoda je využívána při scénáři *CREDUI* a slouží k předdefinovanému vyplnění přihlašovacích údajů. Ty se vezmou s autentizačního balíčku, který je jediným vstupem této funkce. Tyto údaje jsou uloženy v systémovém uživatelském credentil manageru. Jsou to zpravidla lokální účty na síťovém PC. V případě že není metoda podporovaná, CP vrací *E_NOTIMPL*. V opačném případě vrací návratový kód operace s autentizačím balíčkem.

Advise
Implementací této metody CP umožňuje dynamické chování, kdy vstupem funkce je registrovaný callback na událost. Tuto událost vyvolá některá další komponenta. Většinou to bývá jedna z vlastních credential a nebo jiné vlákno, které obsluhuje nějakou hardwarovou komponentu (například čtečku čipových karet).

UnAdvise
Touto metodou dává systém na vědomí, že předchozí callback, registrovaný funkcí *Advise*, je již neplatný. Nemá žádné vstupní parametry.

GetFieldDescriptorCount
Metoda, pomocí které se systém ptá, kolik grafických prvků mají credentials tohoto provideru. Z toho vyplývá předpoklad, že všechny credentials mají stejný počet grafických prvků. Metoda má pouze výstupní parametr, ve kterém CP vrací počet těchto grafických prvků.

GetFieldDescriptorAt
Metoda, kterou systém volá po *GetFieldDescriptorCount*. Získává si informace o grafickém prvku na indexu, který je předán jako první parametr. Druhý parametr této funkce je výstupní informace o id, typu a popisu daného prvku. U textových komponent je tento popis zobrazen po načtení credentialu v podobě zašedlého textu na pozadí, který se automaticky schová při první editaci daného prvku.

GetCredentialCount
Touto metodou si získává systém informace o počtu dlaždic, které by měl následně zobrazit. Funkce nemá žádné vstupní parametry, pouze výstupní. První je počet vlastních credentials, druhý je index základního provideru. Pokud je tento parametr nastaven na

hodnotu *CREDENTIAL_PROVIDER_NO_DEFAULT* a nebo je vrácen index mimo rozsah credentials, žádný credential není považován jako základní. Třetí parametr je příznak, zda se má provést auto-login se základním credentials.

GetCredentialAt

Funkce pro získání pointeru na credential. Je volána po získání počtu credentials. Má dva parametry, první je vstupní – index credentialu a druhý výstupní – pointer na žádaný provider.

SetUserArray

Funkce, pomocí které předá systém seznam uživatelů, pro které vyžaduje jednotlivé credentials. Používá se jen u CP v2.

3.2 Credential

Credential musí implementovat rozhraní *ICredentialProviderCredential* (Výpis 3) a v případě CP v2 rozhraní *ICredentialProviderCredential2* (Výpis 4), které dědí z *ICredentialProviderCredential*. Každý provider si sám spravuje místo v paměti, do kterého se zapisují a čtou údaje, které uživatel nebo systém zapsal do jednotlivých grafických polí.

```
ICredentialProviderCredential : public IUnknown
{
public:
    virtual  HRESULT STDMETHODCALLTYPE Advise(
        /* [in] */ ICredentialProviderCredentialEvents *pcpce) = 0;

    virtual  HRESULT STDMETHODCALLTYPE UnAdvise( void) = 0;

    virtual  HRESULT STDMETHODCALLTYPE SetSelected(
        /* [out] */ BOOL *pbAutoLogon) = 0;

    virtual  HRESULT STDMETHODCALLTYPE SetDeselected( void) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetFieldState(
        /* [in] */ DWORD dwFieldID,
        /* [out] */ CREDENTIAL_PROVIDER_FIELD_STATE *pcpfs,
        /* [out] */ CREDENTIAL_PROVIDER_FIELD_INTERACTIVE_STATE *pcpfis) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetStringValue(
        /* [in] */ DWORD dwFieldID,
        /* [string ][out] */ LPWSTR *ppsz) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetBitmapValue(
        /* [in] */ DWORD dwFieldID,
        /* [out] */ HBITMAP *phbmp) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetCheckboxValue(
        /* [in] */ DWORD dwFieldID,
        /* [out] */ BOOL *pbChecked,
        /* [string ][out] */ LPWSTR *ppszLabel) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetSubmitButtonValue(
```

```
        /* [in] */ DWORD dwFieldID,
        /* [out] */ DWORD *pdwAdjacentTo) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetComboBoxValueCount(
        /* [in] */ DWORD dwFieldID,
        /* [out] */ DWORD *pcItems,
        /* [out] */ DWORD *pdwSelectedItem) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetComboBoxValueAt(
        /* [in] */ DWORD dwFieldID,
        DWORD dwItem,
        /* [string ][out] */ LPWSTR *ppszItem) = 0;

    virtual  HRESULT STDMETHODCALLTYPE SetStringValue(
        /* [in] */ DWORD dwFieldID,
        /* [string ][in] */ LPCWSTR psz) = 0;

    virtual  HRESULT STDMETHODCALLTYPE SetCheckboxValue(
        /* [in] */ DWORD dwFieldID,
        /* [in] */ BOOL bChecked) = 0;

    virtual  HRESULT STDMETHODCALLTYPE SetComboBoxSelectedValue(
        /* [in] */ DWORD dwFieldID,
        /* [in] */ DWORD dwSelectedItem) = 0;

    virtual  HRESULT STDMETHODCALLTYPE CommandLinkClicked(
        /* [in] */ DWORD dwFieldID) = 0;

    virtual  HRESULT STDMETHODCALLTYPE GetSerialization(
        /* [out] */ CREDENTIAL_PROVIDER_GET_SERIALIZATION_RESPONSE *pcpgsr,
        /* [out] */ CREDENTIAL_PROVIDER_CREDENTIAL_SERIALIZATION *pcpcs,
        /* [out] */ LPWSTR *ppszOptionalStatusText,
        /* [out] */ CREDENTIAL_PROVIDER_STATUS_ICON *pcpsiOptionalStatusIcon) = 0;

    virtual  HRESULT STDMETHODCALLTYPE ReportResult(
        /* [in] */ NTSTATUS ntsStatus,
        /* [in] */ NTSTATUS ntsSubstatus,
        /* [out] */ LPWSTR *ppszOptionalStatusText,
        /* [out] */ CREDENTIAL_PROVIDER_STATUS_ICON *pcpsiOptionalStatusIcon) = 0;

};
```

Výpis 3: Rozhraní ICredentialProviderCredential

```
ICredentialProviderCredential2 : public ICredentialProviderCredential
{
public:
    virtual  HRESULT STDMETHODCALLTYPE GetUserSid(
        /* [string ][out] */ LPWSTR *sid) = 0;

};
```

Výpis 4: Rozhraní ICredentialProviderCredential2

Advise

Metoda, která umožňuje credential posílat události jako reakci při zpracování. Toto se používá k dynamickému chování CP. Například při zjištění expirace hesla uživatele je změněn vnitřní stav credential a poslána zpráva systému, aby znovu vyčetl od providera informace o jednotlivých credential. Tato metoda je volána jako první. Jejím jedinným parametrem je pointer na callback.

UnAdvise

UnAdvise je bezparametrová metoda oznamující credential, že dříve obdržený pointer na callback je již neplatný.

SetSelected

Metoda, jejímž prostřednictvím systém oznamuje credential, že byla vybrána. Jediným parametr je výstupní proměnná typu bool oznamující systému, zda je credential typu auto-login a může být použit k přihlášení do systému bez účasti uživatele.

SetDeselected

Bezparametrová metoda oznamující credentialu, že již není vybrána a že by měl vymazat veškeré citlivé údaje, které do něho uživatel vyplnil (heslo, PIN).

GetFieldState

Slouží k získání informací o stavu daného grafického prvku. Jejím jediným vstupním parametrem je index grafického prvku. Druhý parametr je výstupní, který logonu UI říká při kterém stavu se má prvek vykreslit (viz tabulka). Posledním parametr je použit pro určení, zda má být prvek vybrán po prvním načtení. Pokud je tato hodnota nastavena pro více prvků, je zvolen ten s nejnižším číslem indexu.

GetStringValue

Logon UI touto metodou získává informace pro vykreslení obsahu textových polí. První parametr je index grafického prvku, na jehož hodnotu se systém ptá. Druhá hodnota je výstupní textový řetězec formátu Unicode s nulou na konci.

GetBitmapValue

Získání obrázku pro dlaždici. Vstupním parametrem je ID grafického prvku, výstupním je pointer na handle s obrázkem.

GetCheckboxValue

Získání informace o stavu checkboxu. Vstupním parametrem je ID grafického prvku, výstupním je proměnná typu bool. Třetí parametr je Unicode řetězes s popiskem checkboxu.

GetSubmitButtonValue

Získání informace o pozici tlačítka k potvrzení ukončení zadávání údajů. Vstupním parametrem je ID potvrzovacího tlačítka, výstupním je ID prvku, vedle kterého se má tlačítko vykreslit. Z estetických důvodů se nedoporučuje vykreslovat tlačítko vedle obrázku.

GetComboBoxValueCount

Metoda pro získání počtu prvků v comboboxu. Prvním parametrem je ID prvku, druhý parametr je výstupní vracející počet hodnot v comboboxu. Poslední parametr je index prvku, který má být vybrán jako základní. Pokud je hodnota mimo rozsah anebo je pointer nulový, žádný prvek se nevybere a zobrazí se popisek grafického prvku.

GetComboBoxValueAt

Metoda pro získání hodnoty pro naplnění comboboxu. Funkce má dva vstupní parametry, ID grafického prvku a index hodnoty comboboxu, která se má vrátit. Posledním parametrem je výstupní proměnná typu Unicode řetězec obsahující textovou hodnotu.

SetStringValue

Metoda upozorňující credential na změnu textového editačního políčka. Credential si podle toho musí upravit hodnotu uloženou ve své vnitřní proměnné. Funkce má dva vstupní parametry, ID grafického prvku a samotnou hodnotu, která se má aktualizovat.

SetCheckboxValue

Metoda upozorňující credential na změnu stavu checkboxu. Prvním parametrem je ID grafického prvku a druhým bool reprezentace stavu comboboxu.

SetComboBoxSelectedValue

Touto metodou systém upozorňuje na změnu vybraného textového políčka v comboboxu. Funkce má dva parametry, ID grafického prvku a index vybrané hodnoty.

CommandLinkClicked

Metoda reagující na klik na grafický prvek typu *CPFT_COMMAND_LINK*. Jedná se o prvek textového charakteru, který nejde editovat a který umožňuje uživateli spustit nějaký příkaz (například vynucené znovupřipojení k čipové kartě). Jediný parametr je ID grafického prvku.

GetSerialization

Metoda vyvolaná kliknutím na potvrzující tlačítko. Nemá žádné vstupní parametry. První výstupní parametr je stav vytváření autentizačního balíčku, druhým je vlastní autentizační balíček. Další parametr je volitelný textový řetězec s popisem stavu operace a poslední je ikona zobrazená s tímto popisem.

ReportResult

Metoda volaná v případě neúspěchu serializace autentizačního balíčku. Má dva vstupní parametry, hodnotu statusu a substatusu. Funkce vrací dvě proměnné, textový popis chyby nebo důvodu neúspěchu a jeho ikonu.

GetUserSid

Metoda implementovaná jen v rozhraní v2. Jejím jediným parametrem je výstupní hodnota identifikující uživatele, pro kterého byl credential vytvořen.

4 Využití externích karet

Standardní využití čipové karty je dvoufázová autentizace, tzn. uživatel musí fyzicky držet čipovou kartu a znát její pin. Pro využití čipové karty k demonstraci použití uloženého jména a hesla proto stačí použít jednofázovou autentizaci, samotné držení čipové karty.

Jako samotný nosič dostačuje karta MIFARE DESFire, na které se uloží všechny potřebné údaje k autentizaci pomocí hesla a to jsou název domény, uživatelské jméno a heslo. Ochrana přístupu dat se zajistí nastavením omezení přístupu k aplikaci s tím, že se musí provést autentizace s hlavním klíčem karty.

Pro samotné využití karty je potřeba napsat API pomocí knihovny winscard.dll.

4.1 Správa přihlašovacích údajů na kartě

Pro změnu přihlašovacích údajů na kartě lze použít dva principy – externí aplikace, anebo CP se scénářem změny hesla. Použití CP má výhodu v tom, že změna hesla na kartě zároveň provede i změnu hesla v systému. Toto je uživatelsky příjemnější postup. Nevýhoda spočívá v tom, že při neúspěchu zápisu na kartu, či neúspěchu změny hesla v systému, může nastat situace, kdy nesouhlasí heslo na kartě a v systému, a nastává dilema ověření hesla na kartě. Pokud by CP kontroloval stávající heslo na kartě i v systému, jedna z operací neprojde.

Další problém nastává tehdy, kdy uživatel změní heslo mimo tento CP a na kartě se nachází neaktuální heslo. Ve všech těchto případech se musí použít externí aplikace pro změnu údajů, aby sjednotila stav mezi kartou a systémem. Z tohoto vyplývá, že externí aplikace je nutností.

Externí aplikace má výhodu i v možnosti přidávat a ubírat jednotlivé uživatele. Uživatel musí mít jen na paměti, že na kartě je možno uložit jen jeden záznam s kombinací doména a uživatelské jméno. Pokud by držitel karty měl více lokálních účtů se stejným jménem, ale různými hesly, musí použít více karet na uložení svých přihlašovacích údajů. Aplikaci lze použít i v případě bez CP na změnu hesla. Uživatel si ale musí vždy změnit heslo na dvou místech.

Mazání účtu z karty by také mělo ověřovat znalost stávajícího hesla. V případu zapomenutí hesla je potřeba vyvinout speciální verzi aplikace, kterou v reálném použití drží administrátoři, a která umožňuje změnu hesla bez ověření znalosti starého hesla. Tato aplikace představuje bezpečnostní riziko. V případě nasazení se proto musí zvážit bezpečnostní opatření na kontrolu práce, například okamžité auditování všech operací s kartou na síťový auditní server, na který nemá administrátor přístup. S tím je spojena nutnost pravidelné kontroly těchto logů zvoleným auditorem, který ale nesmí být v roli administrátora.

4.2 Credential Provider pro použití čipových karet s uloženými údaji

Přihlašovací data se uloží na čipovou kartu DESFire se speciálně vytvořenou aplikací s AID 0x010000, ve které se vytvoří jeden standardní soubor pro zápis dat, s číslem 01 a s

Tag	Hodnota	Popis
RECORDS_COUNT	0x01	Počet záznamů s přihlašovacími údaji
RECORD_DATA	0x02	Jednotlivé přihlašovací údaje
DOMAIN	0x03	Název domény (pro lokální uživatele prázdné)
USER_NAME	0x04	Uživatelské jméno
PASSWD	0x05	Přihlašovací heslo

Tabulka 4: Tabulka použitých tagů

alokovanou velikosti do 1k, 2k či 4k dle velikosti karty.

Data na kartu se budou ukládat v podobě TLV struktury (viz Tabulka 4).

Všechny prvky musí mít maximální délku dat 255, takže element L bude vždy na jeden bajt. Na prvních třech bajtech dat je tag *RECORDS_COUNT*, který určuje kolik tagů *RECORD_DATA* následuje za ním. Data v tagu *RECORD_DATA* mají vždy délku 255, a jejich obsahem je posloupnost tagů *DOMAIN*, *USER_NAME* a *PASSWD*. Zbytek tagu do velikosti 255 je doplněn 0x00. Díky tomuto uspořádání lze čtení provádět ve dvou příkazech, jedno čtení na získání počtu záznamů a druhé pro získání samotných dat. Velikost dat je 257 bajtů (1B Tag + 1B Length + 255B Data), takže do 2k bajtů velkého souboru můžeme uložit maximálně 7 záznamů, které budou celkem zabírat 1802 bajtů.

Samotný CP bude fungovat ve dvou vláknech. V prvním běží sám provider a jeho credentials a v druhém vláknu běží kód, jehož prací je sledovat připojené čtečky čipových karet a kontrolovat zda se k té čtečce nepřiblíží bezkontaktní karta. V tom případě se provede pokus o přihlášení k aplikaci s AID 0x010000 za použití hlavního klíče. Tento klíč musí být bezpečně uložen, např. v privátním registru CP, a zabezpečen proti zcizení a zneužít. V případě úspěšného vyčtení přihlašovacích údajů z karty se pošle zpráva providerovi.

Dokud provider nedostane zprávu o úspěšném vyčtení přihlašovacích dat, zobrazuje jednoduchý credential s obrázkem a textem žádající uživatele o přiložení čipové karty. Tento credential má GUI prvky ve stavu *IN_DESELECTED_TILE* a implementace události *SetSelected* vrací chybový stav. Tudíž při pokusu o jeho vybrání se nic neprovede.

V případě přijetí notifikace o změně stavu si vyptá provider seznam vyčtených uživatelských údajů a vytvoří příslušný počet objektů credential. Zároveň se schová základní credential se zprávou o přiložení karty. Provider při vytvoření předá jednotlivým credentials jejich přihlašovací údaje.

Credentials zobrazí jako svůj titulek název domény a jméno příslušného uživatelského účtu. Po vybrání credentialu se zobrazí titulek, obrázek a vyplněné políčko s heslem, vedle kterého je tlačítko na potvrzení. Uživateli jen stačí kliknout na potvrzovací tlačítko a přihlásí se do systému. Pokud je heslo na kartě nesprávné, může uživatel smazat předvyplněné heslo a zadat heslo správné.

Souběžně s tím sleduje druhé vlákno stav karty, a pokud se zjistí odejmutí karty, vymažou se vnitřní proměnné s jménem a heslem a pošle se upozornění providerovi o změně stavu. Provider smaže všechny existující credentialy s přihlašovacími údaji a zobrazí jen základní credential s pokynem na přiložení karty. Pokud v době odebrání karty

je některý credential od tohoto providera vybraný, přepne se GUI do stavu se seznamem jednotlivých dlaždic.

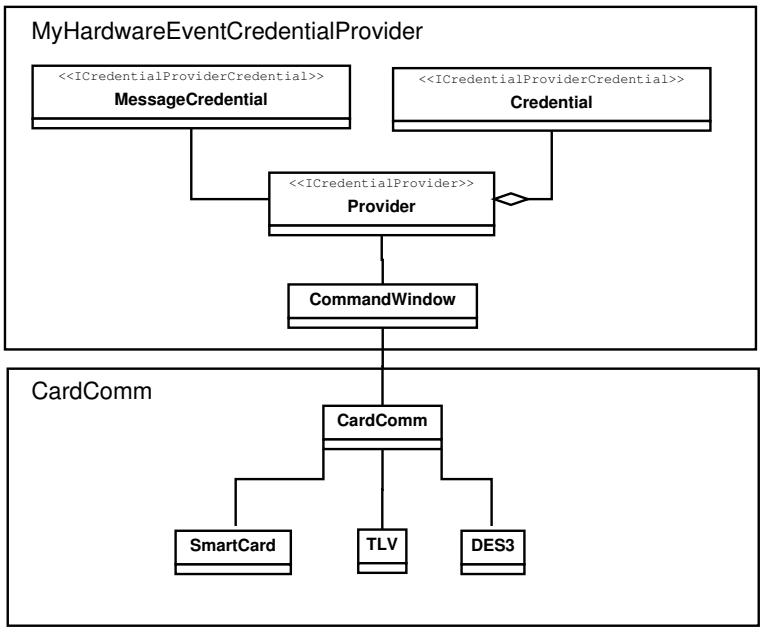
4.3 Implementace Credential Provideru

Credential provider byl implementován podle analýzy s drobnými úpravami. Jako kostra CP byl použit příklad SampleHardwareEventCredentialProvider dodávaný s CP SDK. V rámci pochopení a ověření navrženého konceptu byly implementovány scénáře obou přihlášení uživatele a odemčení sezení. Externí aplikace na správu uživatelských údajů na kartě byla implementována jako konzolová aplikace ve verzi pro Administrátora. Všechny funkce pro uživatele jsou implementovány v této verzi.

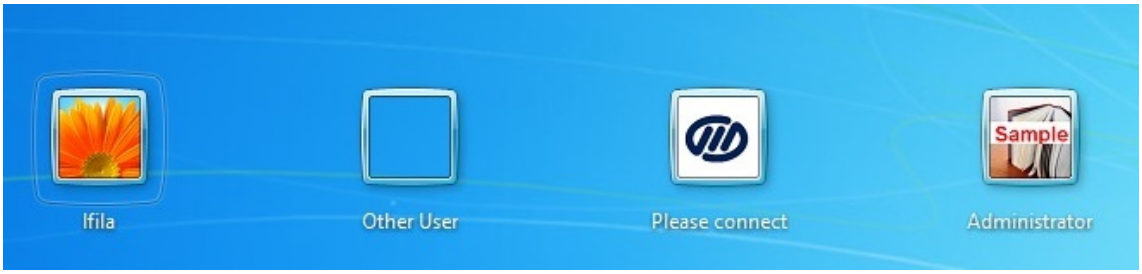
Jednotlivé komponenty CP lze vidět na Obrázku 2. Skládá se ze dvou knihoven. První je CardComm, která zapouzdřuje veškerou komunikaci s kartou, včetně zápisu dat. Skládá se z jedné přístupové třídy *CardComm* a pomocné třídy *SmartCard* pro komunikaci s kartou, *TLV* pro zpracování načtených dat a *DES3*, pro kryptografickou podporu při autentizaci. Druhá komponenta je samotná knihovna CP. Obsahuje jednu třídu implementující rozhraní provideru *Provider*, dvě třídy implementující rozhraní credential (*MessageCredential* a *Credential*) a třídu komunikující s kartou ve vedlejším vlákně *CommandWindow*.

Personalize karet MIFARE DESFire je provedena pomocí skriptů aplikace Multipad. Tyto skripty jsou přiloženy v neveřejné příloze.

Při implementaci CP jsem narazil na několik problémů. Jedním z nich je přetahování se se systémem o přístup k čipové kartě. MS Windows si totiž na pozadí snaží zjistit typ



Obrázek 2: Seznam komponent implementace Credential Provideru



Obrázek 3: Přihlašovací obrazovka - čekání na přiložení karty

karty a dohledat patřičný ovladač. To způsobuje chyby čtení z karty. Proto je po každém neúspěšném vyčtení údajů z karty uzavřeno a znovu vytvořeno sezení pro komunikaci s kartou. CP se snaží vyčíst údaje maximálně 5x, poté se musí karta odebrat a znovu ke čtečce přiložit.

Další častý problem při testování je občasná nemožnost přepsat knihovnu s CP. Při základním nastavení systému bylo občas nutné CP odregistrovat a celý systém restartovat, než bylo možné knihovnu přepsat.

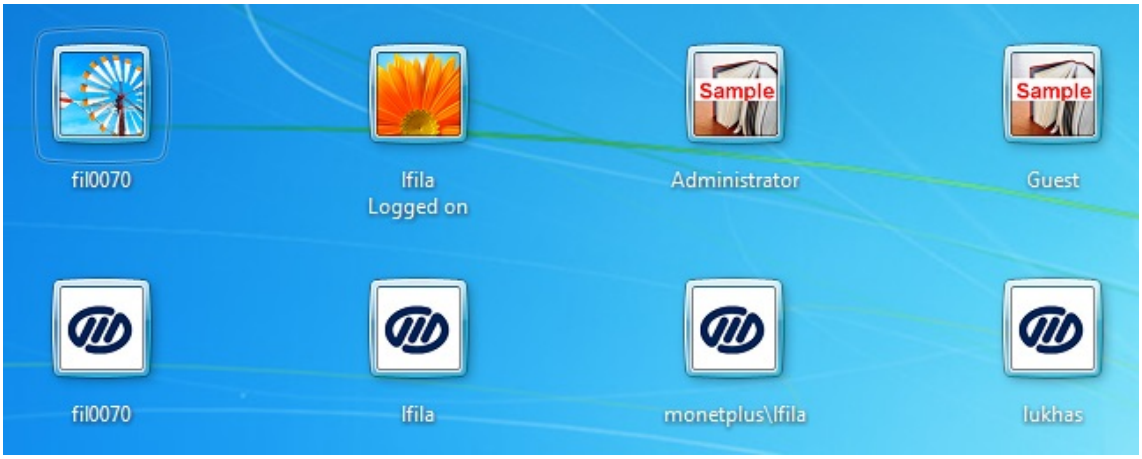
Credential Provider jsem primárně testoval na virtuálním stroji, a to z důvodů bezpečnosti. Při kritické chybě CP shodí celý winlogon.exe a uživatel musí restartovat PC a odregistrovat CP v Safe módu. Pokud se to stane ve virtuálním stroji, je celkové zdržení menší. Použití virtuálního stroje ale nese nepříjemný problém v přebírání přístupu ke čtečce mezi systémem na PC a systémem ve virtuálním stroji. Toto nastává především po připojení čtečky do USB.

Na Obrázku 3 vidíme základní credential od standardního provideru pro přihlášení pomocí jména a hesla. Vedle něho je možnost pro zobrazení všech jeho dlaždic, naši dlaždici informující o stavu komunikace s kartou a poslední dlaždici, která je ukázkou provideru od firmy Microsoft.

Obrázek 4 ukazuje stav po přiložení karty při odemknutí uzamčeného sezení, ve kterém je přihlášen uživatel lfila. První dvě dlaždice jsou od standardního CP. Další dvě v prvním řádku jsou dlaždice z ukázky. Ve spodní řádku jsou vykresleny všechny nalezené přihlašovací údaje z karty. Z nápisu dlaždice je poznat, který účet je lokální a který je pro doménového uživatele.

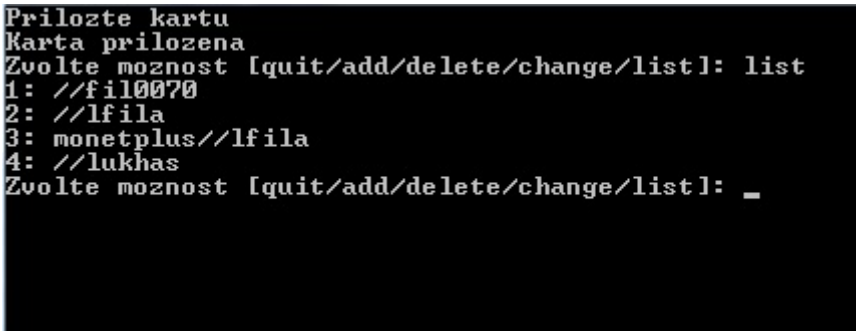
Obrázek 5 ukazuje stav po vybrání dlaždice s aktuálně přihlášeným uživatelem. Heslo je již vyplněno, uživatel ho ale může změnit.

Ukázku externí aplikace na správu uživatelských účtů na čipové kartě můžeme najít na Obrázeku 6.



Obrázek 4: Přihlašovací obrazovka - po přiložení karty

Obrázek 5: Přihlašovací obrazovka - dlaždice po vybrání



Obrázek 6: Správa uživatelských účtů na čipové kartě

5 Přihlášení do systému MS Windows s pomocí certifikátu a klíče mimo čipovou kartu

5.1 Analýza přihlášení za použití certifikátu

Oproti předchozím systémům Windows (XP a starší) jsou požadavky na parametry certifikátu minimální. Použití klíče musí být povoleno pro digitální podpis. Parametry pro smart card logon jsou nastaveny v doménových politikách (Group Policy) celé domény. Tedy teoreticky kterýkoliv certifikát je možno použít pro Smart Card CP.

Při procesu přihlášení systém nejprve vyčte seznam čteček registrovaných v systému. Pomocí funkce CryptAcquireContext, kdy název kontejneru je název čtečky a název provideru je *Microsoft Base Smart Card Cryptographic Service Provider*, se snaží vytvořit sezení s kartou. Pokud se toto nepovede, znamená to, že čtečka není přítomna, anebo v ní není zasunuta karta. Pokud se sezení ustanoví, proces vyčte všechny certifikáty a ověří, zda je to certifikát vydaný vydavatelem, který je nastaven v doménových politikách. Všechny takto ověřené certifikáty se zkopírují do dočasné, bezpečné paměti na počítači. Každý takový certifikát je poté reprezentován v logon UI jednou dlaždicí.

Po aktivaci dlaždice zadá uživatel PIN pro příslušný kontejner. U čipových karet typu minidriver může mít jedna karta víc PINů (maximálně 8), pro každý certifikát zvlášť. Standardně je použit jeden PIN pro celou kartu a druhý PIN, jemuž se říka PUK, pro přístup kontejneru s prvním PINem. Tento PIN společně s názvem čtečky a kontejneru, názvem uživatele a domény je zabalen do KERB_CERTIFICATE_LOGON struktury a poslán LSA k autentizaci.

LSA předá tyto informace Kerberosu a až ten provádí vyčtení privátního klíče z karty a pomocí PKINIT vytvoří přihlašovací token. AD DS ověří token vůči registrovanému certifikátu a uživatelským údajům a vrátí přihlašovací ticket a SID uživatele.

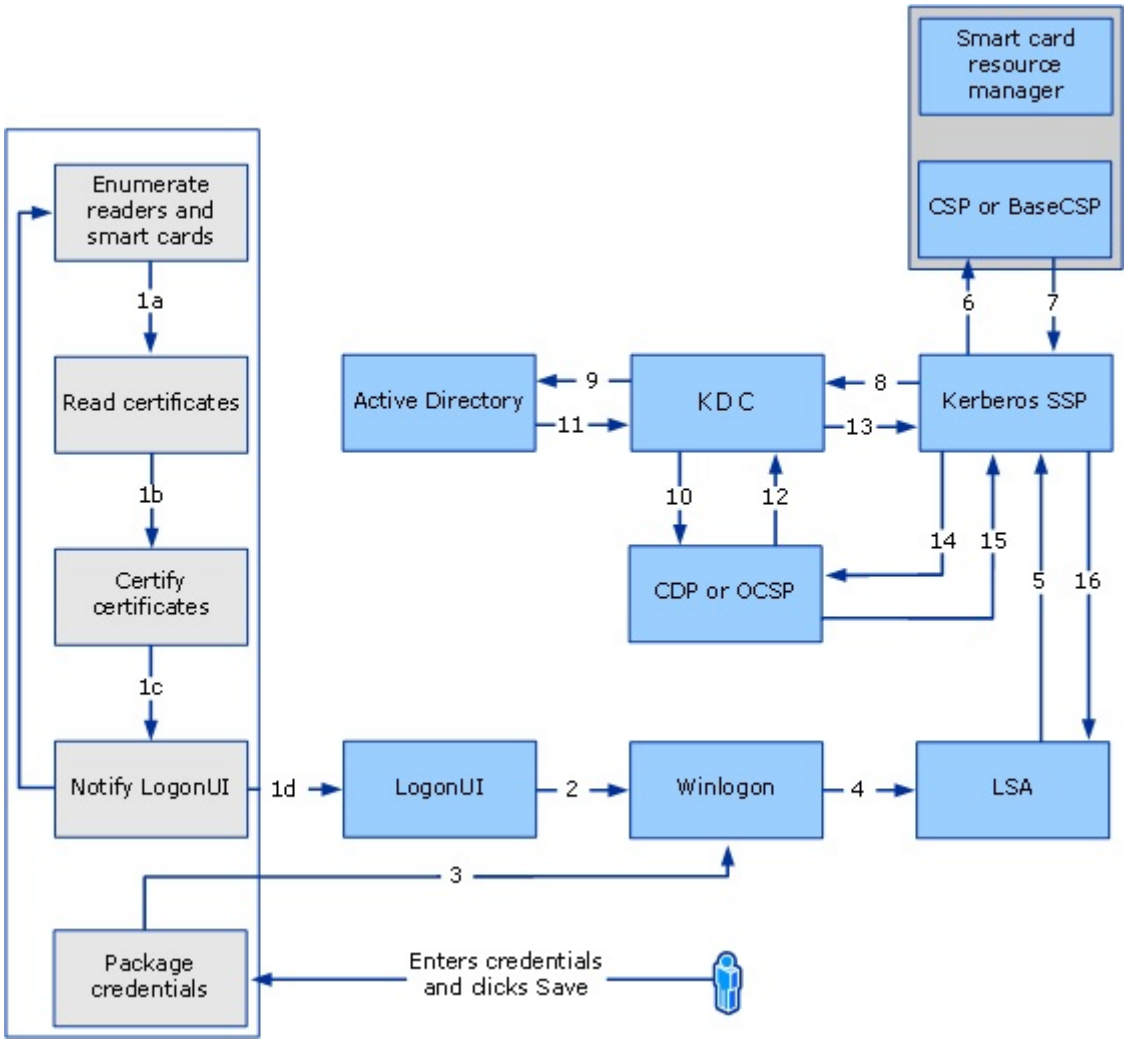
Pokud certifikát tam certifikát ještě není, uloží se do lokálního úložiště certifikátů (MYSTORE) úspěšně autorizovaného uživatele.

Celý proces je znázorněn na Obrázku 7.

Z tohoto principu lze vyvodit, že pokud chceme použít klíč, který není uložen na čipové kartě, musíme splnit několik věcí:

1. V seznamu certifikátů nabízených CP musí být i ty neuložené na čipové kartě.
2. Privátní klíč musí být uložen na místě, ze kterého si ho Kerberos může vyčíst.
3. Privátní klíč nesmí být uložen na čipové kartě.
4. Active Directory musí mít certifikát schválený ve svých doménových politikách.

Pro splnění bodu 1 je potřeba napsat nový CP, který vyčítá certifikáty z jiného uložiště. To může být jedno z úložišť certifikátů na lokálním počítači kromě MYSTORE, protože to je přístupný až po autentizaci uživatele. Další variantou je síťová služba běžící na některém serveru v doméně.



Obrázek 7: Přihlášení do sytému pomocí certifikátu na čipové kartě
(zdroj: http://www.msdn.microsoft.com)

V případě rozsáhlejší infrastruktury s desítkami certifikátů a uživatelů bude potřeba vyžádat od uživatele jeho doménové jméno, aby bylo možné nabízené certifikáty odfiltrovat a uživatel neměl na výběr z desítek dlaždic pro přihlášení.

Splnění bodů 2, 3 a 4 je problematické. V dostupných dokumentací na `msdn.microsoft.com` a různých diskusních fórech na Internetu je uvedeno, že pro autentizaci s vyžití certifikátu používá Kerberos na pokyn od LSA pouze *Microsoft Base Smart Card Cryptographic Service Provider*.

5.2 Problémy při řešení

Při pokusu o ověření funkčnosti jsem pracoval s následujícím předpokladem. Pokud získáme certifikát, který má privátní klíč uložený na čipové kartě a s touto kartou se lze přihlásit do systému (splněn bod 2 a 4), a máme k tomuto certifikátu i privátní klíč v souboru (.pfx), poté by mělo by jít importovat klíč do úložiště klíčů na lokálním počítači, změnit CSP (splněn bod 1 a 3) a přihlásit se takto do systému.

Pro zjednodušení se provede ověření bez použití CP, ale napíše se jednoduchá konzolová aplikace využívající API pro komunikaci s LSA. V tomto případě lze použít lokální úložiště klíčů a certifikátů, které jsou v případě CP nedostupné.

Jako základ pro testovací aplikaci jsem použil ukázku od Mounira Idrassiho [7] pro autentizaci s certifikátem a klíčem na čipové kartě. Pro komunikaci s LSA je použita funkce *LSALogonUser*, která používá podobné vstupní parametry jako ty, které vrací systému CP ve funkci *GetSerialization*.

V prvním kroku jsem ověřil správnost autentizace při použití čipové karty. Tento pokus dopadl úspěšně. Pro další krok bylo nutné zavést údaje z pfx souboru do lokálního úložiště. K tomu jsem využil systémový nástroj certutil.exe. Tato konzolová aplikace se v základním nastavení snaží uložit certifikát a klíč na čipovou kartu. Bylo nutné změnit CSP na takový, který používá softwarové úložiště. Použil jsem *Microsoft Strong Cryptographic Provider* a certifikát úspěšně uložil.

Pokus o přihlášení s tímto certifikátem ovšem dopadl neúspěšně. Po delším průzkumu úložiště jsem objevil, že i když je certifikát uložen, v úložišti klíčů není přítomen jeho privátní klíč. Tato situace byla napravena za použití aplikace *certutil.exe* s CSP *Microsoft Software Key Storage Provider* a klíč byl uložen na požadované místo.

Při další neúspěšném testu a konzultaci s vedoucím, jsem objevil další problém. I když byl certifikát i klíč uložen pomocí správného CSP, nebyli navzájem správně provázáni. Certifikát v sobě obsahoval odkaz na kontejner na čipové kartě, který měl stejný název jak kontejner certifikátu v lokálním úložišti. Na vlastní čipové kartě ležel klíč v kontejneru s úplně jiným názvem. Vedle názvu kontejneru klíče byl špatně i název patřičného přístupového CSP, který byl nastaven na *Microsoft Base Smart Card Cryptographic Service Provider*. Proto jsem napsal program, který přepsal název kontejneru i CSP (viz příloha LogonTest).

Ani tato ruční oprava nepřinesla správný výsledek. Při zkoušení různých standardních CSP od firmy Microsoft se střídaly výsledky, kdy funkce vracela chybu o špatném PINu, anebo o chybějícím klíči. Pro to jsem se rozhodl zkusit klíč vložit do lokálního úložiště pomocí vlastní aplikace využívající CNG API (více příloha cngtest). Po dešifrování a rozdělení ASN struktury získané z pfx souboru jsem hodnoty klíče vložil přímo do kódu a vložil klíč do lokálního úložiště.

Přepis názvu kontejneru a vyzkoušení všech různých CSP nepřineslo očekávaný výsledek.

V posledním pokusu jsem zkusil změnit metodu pro přístup k LSA. Místo funkce *LSALogonUser*, která používá stejné zabalené autentizační balíčky jako CP, jsem použil funkci *LogonUser*. Ta nepoužívá jako vstupní parametry názvy kontejnerů, ale přímo SHA-1 Hash certifikátu, se kterým se daný uživatel snaží přihlásit. Jako základ jsem po-

užil další volně dostupnou ukázku použití od Mounira Idrassiho. Bohužel i tento pokus dopadl neúspěšně.

Z neúspěchu těchto experimentů jsem vyvodil, že zadání této práce na autentizaci do systému Windows za pomocí certifikátu a klíče bez použití čipové karty a s použitím standardních prostředků od firmy Microsoft (CSP a KSP) nelze splnit.

Přesná příčina neúspěchu není jasná. V dokumentacích a oficiálních fórech firmy Microsoft je uvedeno, že při přihlášení pomocí certifikátu je nutná čipová karta. Je tedy možné, že samotný Kerberos vyžaduje CSP, které je buď samotný *Base Smart Card CSP*, a nebo rozšíření třetí strany, které je z objektového hlediska potomkem třídy toho CSP. Druhá myšlenka je ta, že samotné CSP, jež pracují se softwarovým úložištěm, mají v sobě tuto možnost zakázanou. Kvůli nemožnosti prozkoumat zdrojové kódy Kerberosu a příliš obecné dokumentaci, tyto teze nelze potvrdit či vyvrátit.

6 Závěr

Zadání práce bylo rozděleno na tři úkoly. Analýza a porozumění technologie Credential Provider, návrh a implementace Credential Provideru pro použití čipových karet jako úložiště jednotlivých přihlašovacích údajů a návrh a implementace Credential Provideru pro přihlášení za pomocí certifikátu a klíče mimo čipovou kartu.

První úkol byl splněn. Na základě dokumentace a dodaných příkladů od firmy Microsoft byla provedena analýza rozhraní, které musí všechny výsledné CP implementovat. Ověření správnosti těchto poznatků bylo provedeno v druhém úkolu.

Úkol číslo dvě se také podařilo splnit, jeho výsledkem je DLL knihovna, funkčně demonstrující spolupráci Credential Provideru s hardwarovým zařízením. Pro případné nasazení knihovny do reálného provozu je nutné vyřešit problém s distribucí, uložením a použití hlavního klíče karty. V praxi je nutné klíč diverzifikovat podle UID, aby při kompromitaci klíče z jedné karty nebyly kompromitovány všechny karty. Případně je možné používat pro přístup k aplikace klíč aplikační. Doporučil bych uložit hlavní klíč na bezpečný hardware, například HSM modul, a důvěryhodným klientům poskytnout kryptografickou podporu.

Poslední úkol byl splněn jen z části. Nebyla nalezena možnost, jak s pomocí standardních prostředků systému Windows provést přihlášení do systému pomocí certifikát s klíčem v softwarovém úložišti.

Samotné přihlášení bez čipové karty lze vyřešit několika způsoby. Jedno řešení leží v napsání vlastních kryptografických poskytovatelích pro práci s čipovou kartou, kteří budou dědit z Base Smart Card CSP, ale při práci s privátním klíčem pracuje s softwarovým úložištěm. Druhé řešení spočívá v implementaci vlastního CSP, které pracuje s klíči a certifikáty v softwarovým úložišti. Obě tyto varianty vyžadují, aby byly použity s nestandardním CP.

Další řešení spočívá v napsání ovladače pro virtuální čtečku karet, která se bude chovat jako každá jiná PC/SC čtečka, ale bude přistupovat do softwarového úložiště. Toto řešení má výhodu v tom, že je možné bez problému využít stávající moduly od firmy Microsoft (CP, CSP i KSP) pro použití čipových karet.

Realizace jednoho z výše uvedených návrhů na řešení by mohlo být pokračování této práce. Implementace vlastního CSP či ovladače je netriviální a spolu s propojením na autorizační server CASE a zapouzdření do projektu virtuální karty je vhodným tématem Diplomové práce.

7 Reference

[1] Griffin, Dan, *Create Custom Login Experiences With Credential Providers For Windows Vista*, online, 2007 [cit. 3.5. 2015], url: https://msdn.microsoft.com/cs-cz/magazine/cc163489%28en-us%29.aspx.

[2] Micrsoft, *Windows Vista Smart Card Infrastructure*, online [cit. 3.5. 2015], url: https://msdn.microsoft.com/en-us/library/bb905527.aspx.

[3] Micrsoft, *MSDN library*, online [cit. 3.5. 2015], url: https://msdn.microsoft.com/.

[4] Dostálek, Libor a kolektiv, *Velký průvodce protokoly TCP/IP - Bezpečnost*, Computer Press, 2003.

[5] NXP Semiconductors *MIFARE DESFire EV1 contactless multi-application IC, Rev. 3.1*, online, 2010 [cit. 3.5. 2015], url: http://www.nxp.com/documents/short_data_sheet/MF3ICDX21_41_81_SDS.pdf.

[6] Micrsoft, *Smart card logon flow in Windows Vista and Windows 7*, online [cit. 5.5. 2015], url: https://technet.microsoft.com/en-us/library/92aea8c5-c617-4a04-9356-e0ad8dd4cea9%28v=ws.10%29#BKMK_SmartCardLogonFlowVista.

[7] Idrassi, Mounir, *LsaSmartCardLogon example2*, online [cit. 3.3. 2015], url: www.idrix.fr/Root/Samples/LsaSmartCardLogon2.cpp.

A Zdrojové soubory

Neveřejné zdrojové soubory jsou na přiloženém CD.

- **CardComm** - C++ knihovna, komunikace s čipovou kartou včetně autentizace a práce s TLV daty
- **CardCommTest** - C++ konzolová aplikace, testování komunikační knihovny
- **cngtest** - C++ konzolová aplikace, vložení klíče do keystore
- **enumkey** - C++ konzolová aplikace, vyčítání informací o klíčích z keystore
- **LogonTest** - C++ konzolová aplikace, testování přihlášení pomocí funkce LogonUser
- **LsaLogon** - C++ konzolová aplikace, testovaní přihlášení pomocí funkce LsaLogonUser
- **Multipad** - multipad skripty, sada skriptů pro personalizace, mazání, zapis a čtení MIFARE Desfire karty
- **MyHardwareEventCredentialProvider** - C++ knihovna, CP pro přihlášení uživatelů s přihlašovacími údaji na čipové kartě
- **UserControl** - C++ konzolová aplikace, správa uživatelských účtů na čipové kartě